Smart Contract Security Audit

SAKTOKEN

AUDIT CITY received the smart contract security audit of SAKTOKEN on January 22, 2022. The following are the details and results of this smart contract security audit:

Token Name: SAKTOKEN **$SAK**

Contract address: 0xef2284e93c74a1b6b024b3b45d71858f5fdd128e

Link Address:

https://bscscan.com/token/0xef2284e93c74a1b6b024b3b45d71858f5fdd128e

The audit items and results:

(Other unknown security vulnerabilities are not included in the audit responsibility scope)

Audit Result: Passed

Ownership: Not renounced

(The contract contains ownership functionality and ownership is not renounced which allows the creator or current owner to modify contract behavior)
(Owner can't modify fees)

KYC Verification: Not verified

Audit Number: AUC0028012022
Audit Date: January 25, 2022

Audited By AuditCity

Table of Content

Introduction

This Audit Report mainly focuses on the overall security of SAKTOKEN Smart Contract. With this report, we have tried to ensure the reliability and correctness of their smart contract by complete and rigorous assessment of their system's architecture and the smart contract codebase.

Auditing Approach and Methodologies applied

The AUDIT CITY team has performed rigorous testing of the project starting with analyzing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third-party smart contracts and libraries.

Our team then performed a formal line by line inspection of the Smart Contract to find any potential issue like race conditions, transaction-ordering dependence, timestamp dependence, and denial of service attacks.

In the Unit testing Phase, we coded/conducted custom unit tests written for each function in the contract to verify that each function works as expected.

In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was tested in collaboration of our multiple team members and this included -

- Testing the functionality of the Smart Contract to determine proper logic has been followed throughout the whole process.
- Analyzing the complexity of the code in depth and detailed, manual review of the code, line-by-line.
- Deploying the code on testnet using multiple clients to run live tests.
- Analyzing failure preparations to check how the Smart Contract performs in case of any bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analyzing the security of the on-chain data.

Audit Details

Project Name: **SAKTOKEN**
Website: https://www.saktoken.com/
Platform: Binance Smart Chain
Type of Token: BEP20

Languages: Solidity (Smart contract)
Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Mythril, Contract Library

Audit Goals

The focus of the audit was to verify that the Smart Contract System is secure, resilient and working according to the specifications. The audit activities can be grouped in the following three categories:

Security

Identifying security related issues within each contract and the system of contract.

Sound Architecture

Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.

Code Correctness and Quality

A full review of the contract source code. The primary areas of focus include:

- Accuracy
- Readability
- Sections of code with high complexity
- Quantity and quality of test coverage

Issue Categories

Every issue in this report was assigned a severity level from the following:

High level severity issues

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

Medium level severity issues

Issues on this level could potentially bring problems and should eventually be fixed.

Low level severity issues

Issues on this level are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Issues Checking

Critical	High	Medium	Low	Note
0	0	0	0	0

№	Issue description.	Checking status
1	Compiler warnings.	Passed
2	Race conditions and Reentrancy. Cross-function race conditions.	Passed
3	Possible delays in data delivery.	Passed
4	Oracle calls.	Passed
5	Front running.	Passed
6	Timestamp dependence.	Passed
7	Integer Overflow and Underflow.	Passed
8	DoS with Revert.	Passed
9	DoS with block gas limit.	Passed
10	Methods execution permissions.	Passed
11	Economy model.	Passed
12	The impact of the exchange rate on the logic.	Passed
13	Private user data leaks.	Passed
14	Malicious Event log.	Passed
15	Scoping and Declarations.	Passed
16	Uninitialized storage pointers.	Passed
17	Arithmetic accuracy.	Passed
18	Design Logic.	Passed
19	Cross-function race conditions.	Passed
20	Safe Zeppelin module.	Passed
21	Fallback function security.	Passed

Manual Audit

Manual Audit:

For this section the code was tested/read line by line by our developers. We also used Remix IDE's JavaScript VM and Kovan networks to test the contract functionality.

Critical Severity Issues

No critical severity issues found.

High Severity Issues

No high severity issues found.

Medium Severity Issues

No medium severity issues found.

Low Severity Issues

No low severity issues found.